Alerts Software Version 1.3.4.2

for Solaris 2.5.1

Version Description Document

25 April 1997

Program Manager
Common Hardware Software
Fort Monmouth, NJ 07703

Alerts Software Version Description Document V1.3.4.2

CONTENTS

SECTION 1 SCOPE

1.1 IDENTIFICATION

This document Alerts Software Version 1.3.4.2. This version contains corrections to Version 1.3.4.1 See Section 3.10 for more details.

Alerts Version 1.3.4.2 is hosted on the following platforms:

Hardware	Operating System	Ada Compiler	C Compiler
Sun SPARC 20	Solaris 2.4/2.5.1	SPARCWorks Ada 2.1.1	GNU C Compiler

This VDD describes the delivery of the Alerts Services 1.3.4.2 to the Common Software Library. This version of Alerts does not have security capabilities that is needed to run sensitive application programs. Future versions of Alerts will be using the Kerberos, Generic Security Services (GSS) Application Program Interface (API) to satisfy the security shortcomings of Alerts 1.3.4.2

1.2 SYSTEM OVERVIEW

The Alert Software provides a standard way for user applications to initiate and manage Alerts.

Alerts are logically divided by several criteria.

Interactive alerts are that set of alerts requiring operator responses to be returned to their initiating processes or forwarded to other specified processes. Non-interactive alerts are displayed as information only and merely require operator acknowledgment.

Local alerts are those alerts which are relevant only at the physical workstation where they occur, whereas role-specific alerts may be distributed to one or more workstations where a particular activity is being performed.

Persistent alerts are those alerts which have been identified by the client to survive system shutdown. Persistent alerts are reactivated upon startup.

Alerts may be grouped by the client, using UNIX regular expressions. This grouping is implemented in the Alert_ID_Pattern. For example, the expression [A*] represents all alerts whose identifiers begin with A; [*001] , represents all alerts whose identifiers end with 001; (For further information on UNIX regular expressions, consult the UNIX MAN pages.)

An operator interface consisting of a default non-interactive alert display window and default alerts management windows is incorporated into the alerts block.

Alert block clients may interface at the service level or may allow the alert block windows to perform all of the display processing. In the former case, clients must register display processes to be notified upon arrival of relevant alerts.

The Alert Block contains three Computer Software Components (CSC), The Alert Client Operations Software component, the Alert Display Operations Software component, and the Alert Management Operations Component. Each CSC contains Computer Software Units (CSU) which function as APIs. The Alert Client Operations Software APIs include client operations to send and delete alerts, and to receive responses from alerts. The Alert Display Operations Software APIs include client operations to receive, display, respond to, and delete alerts. The Alert Management Operations component contains client operations to suspend and resume notification of local alerts, to enable and disable the ability to suspend notification, and to enable and disable all processing of non-critical alerts.

Functions are available to convert the alert strings from the client format into readable formats for the Alerts Software and to convert the alert strings from the Alerts Software format into formats which are readable to the client.

1.3 DOCUMENTATION OVERVIEW

The purpose of this document is to identify and describe the current version of the Alerts Software. This document also provides the information necessary to install and execute the software. The document is organized into the following sections:

SECTION 1	SCOPE provides the scope of the Version Description Document.
SECTION 2	REFERENCED DOCUMENTS lists the documents referenced in this document.
SECTION 3	VERSION DESCRIPTION lists the following information regarding the release of the Alerts Software: materials released, CSCI contents, adaptation data, bibliography of reference documents, installation instructions, and known problems/errors.
SECTION 4	NOTES defines acronyms and abbreviations used in this document.

SECTION 2 REFERENCED DOCUMENTS

2.1 GOVERNMENT DOCUMENTS

DoD-STD-2167A	Defense System Software Development, 29 February 1988
DI-MCCR-80013A	Data Item Description - Version Description Document, 29 February 1988.

2.2 NON-GOVERNMENT DOCUMENTS

OSF DCE

SUN SPARC 20 Solaris 2.5.1 Operating System Documents

SUN SPARCWorks Ada 2.1.1 Documents

SECTION 3 VERSION DESCRIPTION

3.1 INVENTORY OF MATERIALS RELEASED

3.1.1 SOFTWARE

This Version of the Alerts Software is provided on a 8mm DAT tape in segmented form. The Alerts Software is made up of source, executables, and scripts. The executables are compiled and linked from both Ada code, C, and ANSI "C" code. The scripts are UNIX files that provide a user interface to the Alerts Software files. A breakdown of all code used in the generation of the Alerts Software is listed below. There is also a directory containing source and executable code for unit test software in:

- Ada/C Source Code (src directory)
- ANSI "C " Source Files (src directory)
- Ada Libraries (lib directory)
- Object Files (lib directory)
- Executable Programs (bin directory)
- Build scripts (script directory)
- Test Source Code (src/test_src)

3.1.2 DOCUMENTATION

The following documents are included with this release of the S/W:

Application Program Interface (API) Document for the Alerts Block which includes a new Appendix - APPENDIX 8 - Installation Guide.

All information necessary to load and execute this release of the S/W is provided in the included documentation.

3.2 INVENTORY OF CSCI CONTENTS

The Alerts Software is provided in the form of executable program, source, and scripts. The following is a list of each directory and the files it contains:

- src directory

ada_c_interface.adb
ada_c_interface.ads

ada_signal_handlers.adb
ada_signal_handlers.ads

alert_client_operations.adb
alert_client_operations.ads
alert_client_operations.c

alert_commands.adb
alert_commands.ads

alert_display_operations.adb
alert_display_operations.ads
alert_display_operations.c

alert_internal_types.ads
alert_internal_types.h

alert_management_operations.adb
alert_management_operations.ads
alert_management_operations.c

alert_notification.adb
alert_notification.ads

alert_queue_manager.adb
alert_queue_manager.ads

alert_server_connection.adb
alert_server_connection.ads
alert_server_connection.c

alert_server_tasks.adb
alert_server_tasks.ads

alert_time.adb
alert_time.ads
alert_time.c

alert_type_conversion.adb
alert_type_conversion.ads

alert_types.ads

alert_utilities.adb
alert_utilities.ads
alert_utilities.c

alerts.h

alerts_db.adb
alerts_db.ads

alerts_server.ada

booch_nonpriority_queue.adb
booch_nonpriority_queue.ads

booch_priority_queue.adb
booch_priority_queue.ads

canned_data.c

cass_pipe_io.adb
cass_pipe_io.ads

cass_sockets_accept.ada
cass_sockets.adb
cass_sockets.ads
cass_sockets.c
cass_sockets.connect.ada
cass_sockets.connect_ip.ada
cass_sockets.get.ada
cass_sockets.listen.ada
cass_sockets.put.ada

cass_standard.adb
cass_standard.ads

test_c_alert_client.c
test_c_display_client.c

match.c

prologue

semaphore_pkg.adb
semaphore_pkg.ads

server_shutdown.ada

socket_h.adb
socket_h.ads

unix_services.adb
unix_services.ads

- lib directory

alert_client_operations.o
alert_display_operations.o
alert_management_operations.o
alert_server_connection.o
alert_time.o
alert_utilities.o
canned_data.o
cass_sockets.o
match.o
sunnodce.lib

- lib/nodce_MT.lib directory

GVAS_table
ada.lib
gnrx.lib

- lib/test.lib

GVAS_table
ada.lib
gnrx.lib

- script directory

build_drivers_c
build_server

- db directory

- bin directory

alerts_server
run_alerts
run_ctest_alert
run_ctest_display
run_test_alert
run_test_display
server_shutdown
test_c_alert_client
test_c_display_client
test_alert_client
test_display_client
up

3.3 DISCREPANCY REPORT SUMMARY

This versions contains no problem fixes, only enhancements.

3.4 ADAPTATION DATA

This version of the Alerts Software is delivered to the Common Software Library on one tape, containing the Solaris 2.4/2.5.1 version for the SPARC platform.

3.5 NOT APPLICABLE

3.6 INTERFACE COMPATIBILITY

This version of the Alerts Software removes the dependency on DCE to store the location of the alerts_server process on the LAN.

3.7 NOT APPLICABLE

3.8 RELEASE NOTES

The Alerts_Server procedure has been modified to no longer use DCE's Cell Directory Service (CDS) to obtain the IP address of the running alerts_server process. Instead, it gets the host name of the machine from the environment variable ALERTS_HOST.

The existing Library Units affected by this change are:

Alerts_Server (main procedure)
Alert_Server_Connection

3.9 INSTALLATION INSTRUCTIONS (Non-Segmented Format Only)

3.9.1 REQUIRED

- UNIX

3.9.2 UNLOAD THE TAPE (Non Segmented Format)

The Alerts Block was written to tape in tar format using a relative path name.

To unload the tape create the directory where you want to store the alerts software and extract the software from the tape. For example, to unload the software into a directory called /coe/appl/alerts, you would execute the following commands:

```
%mkdir /coe/appl/alert
%cd /coe/appl/alert
%tar xvf /dev/rmt/0m
```

3.9.3 SETTING UP /ETC/SERVICES

Setting up the etc/services file requires superuser privilege. An entry must be added to the /etc/services file on all nodes where the Alerts Block is to run. The entry should look like:

```
ALERTS      8001/tcp alerts_server
```

The port number does not have to be 8001, but needs to be the same on all nodes using Alerts services.

3.9.4 SETTING UP THE SCRIPTS

There are shell scripts to run the alerts server and the alerts display. These scripts are in the bin subdirectory of alerts. The run_alerts script runs the alerts_server and the ad script runs the alerts displayer. These scripts must be edited so that the environment variables match the local configuration. The following environment variables need modification:

ALERTS	The path to the alerts software.
ALERT_DB	The name of the alerts database. It is currently set to $ALERTS/db/alert_db.
ALERT_HOST	The host name of the machine on which the server is running.

There are scripts in the Alerts script subdirectory used for building the Alerts executables. These scripts must be edited to change the default directory paths to those in the local configuration.

3.9.5 RUNNING THE ALERTS_SERVER

To run the alerts_server, execute the following command:

```
%$ALERTS/run_alerts
```

To shut down the alerts_server, execute:

```
%$ALERTS/bin/server_shutdown
```

3.10 CONCERNS

Version 1.3.4.2 of the Alerts software contains modifications to Version 1.3.4 which prevent the possibility of erroneous results when running multiple Alert Clients. Specifically, semaphores were used to protect data structures from possible update by more than one client. This situation occurred when running many clients, and each client generating alerts simultaneously.

Alerts segment version 1.3.4.2 does not implement certain security features required for some applications. In particular it does not call the server using the GSS (generic security services) API but uses instead standard ports. It is anticipated that future versions of Alerts will use the GSS and Kerberos to implement security.

3.10.1 CURRENT STORAGE REQUIREMENTS

The application has the following storage requirements on the SUN SPARC 20:

- Memory: 2.5 MB

- Storage: 0.717 MB

SECTION 4 NOTES

4.1 ACRONYMS AND ABBREVIATIONS

ANSI	American National Standards Institute
ATCCS	Army Tactical Command and Control System
COE	Common Operating Environment
CDS	Cell Directory Service
CSCI	Computer Software Configuration Item
DCE	Distributed Computing Environment
GSS	Generic Security Services
S/W	Software

The Layer 3 address convention in the TCP/IP world uses a 32 bit binary number to logically identify each node
on the network. The communicating nodes are referred to as hosts and the 32 bit binary address number is the
IP Address. This 32 bit number is represented by breaking the 32 bits into four groups of eight bits each and
representing each eight bit byte with the decimal value equivalent of the binary number.